WHITEPAPER

Intel Corporation

NFVi Benchmarks on HPE* ProLiant* DL380 Gen10 Server with Intel® Xeon® Scalable Processors

Authors

Edith Chang
Hewlett Packard Enterprise (HPE)

Sarita Maini
Intel Corporation

Brad Chaddick
Intel Corporation

Shivapriya Hiremath
Intel Corporation

Xavier Simonart
Intel Corporation

Key Contributors

Al Sanders
Hewlett Packard Enterprise (HPE)

Lee Roberts
Hewlett Packard Enterprise (HPE)

1.0 Introduction

High speed packet forwarding is critical to network functions virtualization infrastructure (NFVi), along with other attributes such as cost, security, and expandability. This white paper discusses how HPE* ProLiant* DL380 Gen10 servers deliver up to 28 percent higher small-packet throughput, compared to the prior Gen9 generation servers. These improvements are the result of various hardware enhancements and software optimizations, which are described in detail in this paper.

The performance testing was conducted by HPE on Gen10 and Gen9 servers using the Yardstick/Network Services Benchmarking (NSB) PROX test framework to run the RFC2544 L3 Forwarding test to benchmark NFVi workloads with Open vSwitch* (OVS) and the Data Plane Development Kit (DPDK). See the "Quick Start Guide for Running Yardstick*/NSB for NFVI Characterization document" for details.

The optimization techniques utilized to achieve optimal performance with HPE ProLiant DL380 servers are provided in the following sections, enabling readers to replicate the performance benchmark results and ultimately develop their own high performance NFVi.

Table of Contents

Tables

Figures

2.0 Test Framework, Test Cases, and Benchmarks

2.1 Network Services Benchmarking (NSB) Test Framework

The Yardstick/Network Services Benchmarking (NSB) test framework provides Communications Service Providers (CoSPs) with common standards and industry-accepted benchmarks for conformance to carrier-grade requirements. Intel contributed its NSB project, along with contributions from industry partners, to the Open Platform for NFV (OPNFV) community. The NSB framework features were added to the Yardstick tool to support the characterization of both NFVi and virtualized network functions (VNFs).

NSB is a benchmarking and characterization tool used to automate NFVi characterization and find performance bottlenecks. It provides deterministic and repeatable benchmarks, and presents metrics in a unified GUI. For this performance study, Yardstick/NSB ran a special test VNF (called PROX), which implements a suite of test cases and displays the benchmarks of the test suite on a Grafana GUI dashboard, which shows the key metrics.

2.2 Intel® Xeon® Scalable Processors

This paper shows the performance gains of the HPE ProLiant DL380 Gen10 servers, compared to Gen9 servers. The Gen10 servers under test are built with Intel® Xeon® Gold 6152 processors (from the Intel Xeon Scalable Processor family, codename Skylake-SP), which have architectural enhancements in the processor and CPU cores, higher core count, larger capacity memory, increased number of PCIe lanes, higher memory bandwidth, higher I/O bandwidth and higher inter-socket bandwidth, compared to the Intel Xeon processors E5-2695 v4 (from the Intel Xeon E5-2600 v4 product family, codename Broadwell-EP) employed in Gen9 servers (see Table 1).

Software	Intel® Xeon® Gold 6152 Processor (Gen10 Server)	Intel® Xeon® Processor E5-2695 v4 (Gen9 Server)
Number of Processor Cores	22	18
Processor Base Frequency	2.10 GHz	2.10 GHz
Maximum Intel® Turbo Boost Max Technology Frequency	3.70 GHz	3.30 GHz
Memory Type/ Maximum Memory Speed	DDR4 @ 2666 MHz	DDR4 @ 2400 MHz
Maximum Number of Memory Channels	6	4
Maximum Number of PCI Express Lanes	48 lanes per CPU	40 lanes per CPU
Point to Point Links	3 Intel UPI Links	2 Intel QPI Links
Mid-Level Cache	1 MB per core	256 KB per core
Last-Level Cache	30.25 MB (non-inclusive)	45 MB (inclusive)

Table 1. Intel® Processor Comparison

2.3 Test Cases

The L3 forwarding throughput of the two generations of HPE ProLiant DL380 servers was evaluated for two configurations using the RFC2544 methodology. The test cases showcase typical NFV-based deployments with the DPDK-accelerated Open vSwitch (OVS-DPDK) and two virtual machines (VMs) performing Layer 3 routing. Measurements were recorded for a frame loss of 0 percent. Figure 1 shows the test configuration, which is meant to simulate a service chaining flow path. Intel® Hyper-Threading Technology (Intel® HT Technology) was disabled and Intel® Turbo Boost Technology was enabled in the BIOS. Additional test configuration information is listed in Table 2.

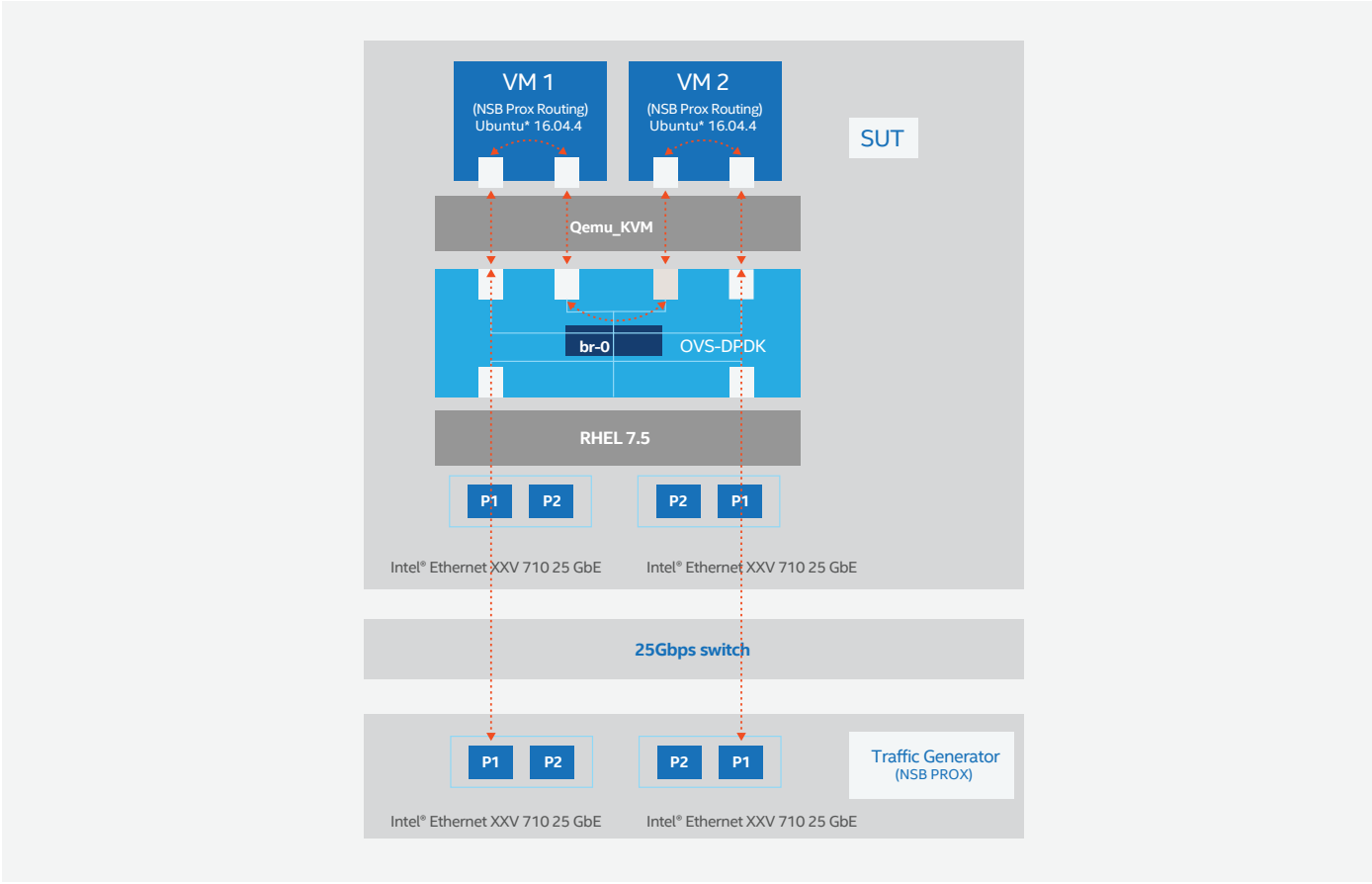

Figure 1. Test Configuration – Phy-VM-VM-Phy

Test Configuration	Number of Virtual CPUs (vCPU) per VM	Core Pinning Schema for OVS Poll Mode Driver (PMD) Threads		Number of Queues	Number of Flows Created Per Port	Number of Aggregate Flows Created
		Number of PMD Threads	All PMD Threads on Different Physical Cores?			
#1 (Phy-VM-VM-Phy)	3	2	Yes	1	8	16
#2 (Phy-VM-VM-Phy)	3	4	Yes	2	8	16

Table 2. Test Configuration Details

2.4 Performance Benchmarks

Figures 2 and 3, and Tables 3 and 4 show the measured L3 forwarding throughput of the HPE ProLiant DL380 Gen10 servers and Gen9 servers for the two previously described test configurations. The Gen10 server delivers up to 28 percent higher L3 forwarding throughput than the Gen9 server. The average throughput was computed as the average of five test runs for each configuration.

Security updates were installed on the servers and the VMs prior to performance measurements.

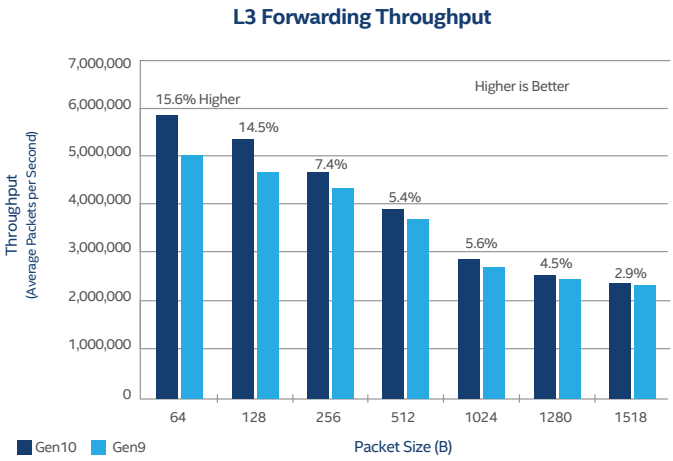

Figure 2. Configuration #1 Test Results (One Queue, Two PMDs, 0% Packet Loss)

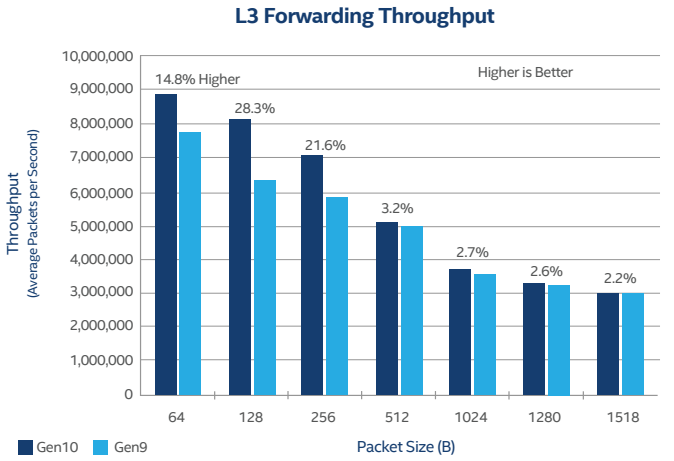

Figure 3. Configuration #2 Test Results (Two Queues, Four PMDs, 0% Packet Loss)

	Gen 10			Gen 9			
Packet Size (B)	Average Packets per Second (PPS)	Average Megabits per Second (Mbps)[1]	Average Throughput (%)	Average PPS	Average Mbps	Average Throughput (%)	**Gen10 vs. Gen9 Improvement (%)**
64	5,837,092	3922	7.85	5,049,933	3393	6.79	**15.59**
128	5,333,702	6315	12.63	4,660,117	5517	11.04	**14.45**
256	4,629,236	10221	20.44	4,312,267	9521	19.04	**7.35**
512	3,893,095	16569	33.14	3,694,233	15722	31.45	**5.38**
1024	2,839,343	23714	47.43	2,689,288	22460	44.92	**5.58**
1280	2,547,826	26497	52.99	2,439,059	25366	50.73	**4.46**
1518	2,374,471	29215	58.43	2,307,668	28393	56.79	**2.89**

Table 3. Configuration #1 Test Results (PHY-VM-VM-PHY, One Queue, Two PMDs, 0% Packet Loss)

[1]The megabits per second (Mbps) metric was calculated using the formula: Mbps = (packet-size + 20) * packets-per-sec * 8 /1,000,000

	Gen 10			Gen 9			
Packet Size (B)	Average Packets per Second (PPS)	Average Megabits per Second (Mbps)	Average Throughput (%)	Average PPS	Average Mbps	Average Throughput (%)	**Gen10 vs. Gen9 Improvement (%)**
64	8,915,783	5991	11.98	7,764,834	5217	10.44	**14.82**
128	8,155,039	9655	19.31	6,358,626	7528	15.06	**28.25**
256	7,087,729	15649	31.30	5,831,068	12874	25.75	**21.55**
512	5,101,391	21711	43.42	4,942,378	21034	42.07	**3.22**
1024	3,684,194	30770	61.54	3,588,118	29967	59.94	**2.68**
1280	3,316,594	34492	68.99	3,232,928	33622	67.24	**2.59**
1518	3,030,393	37285	74.57	2,964,443	36474	72.95	**2.22**

Table 4. Configuration #2 Test Results (PHY-VM-VM-PHY, Two Queues, Four PMDs, 0% Packet Loss)

3.0 Hardware and Software Components

Figure 4 shows the hardware and software stack for HPE ProLiant DL380 Gen10 server based on the Intel Xeon Gold 6152 processor running Red Hat* Enterprise Linux* (RHEL) Server release 7.5 (Maipo)

Table 5 provides additional details about the hardware and software configuration for the HPE ProLiant DL380 Gen10 and Gen9 servers.

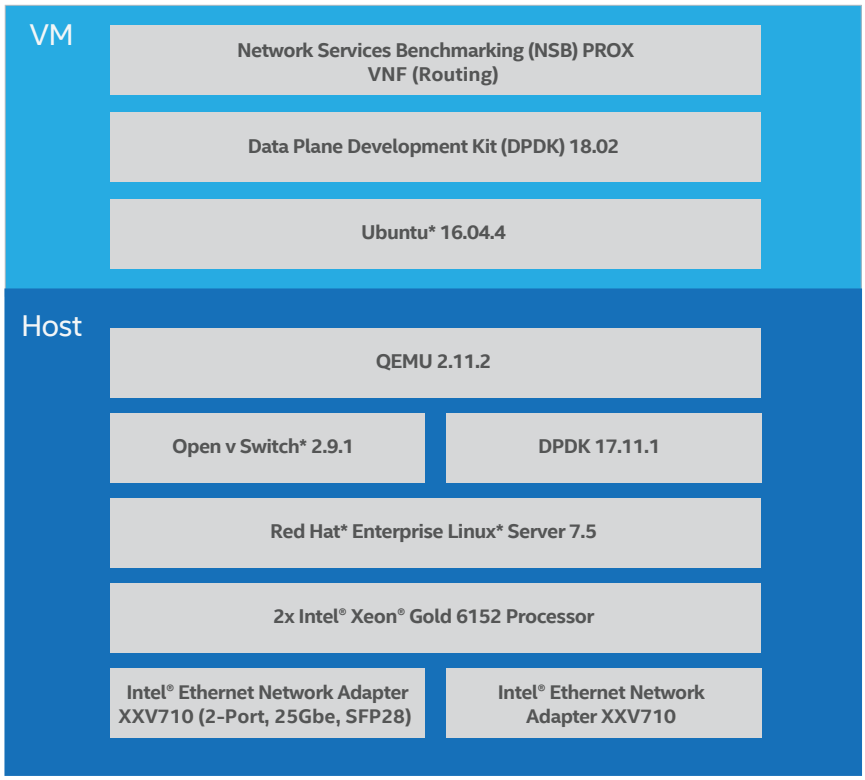

Figure 4. HPE* ProLiant* DL380 Gen10 Server Configuration

Hardware	HPE* ProLiant* DL380 Gen10 Server	HPE* ProLiant* DL380 Gen9 Server
Processor	•2x Intel® Xeon® Gold 6152 Processor, 2.10 GHz. •22 Cores per Socket, Total of 44 Physical Cores, 88 Logical Cores with Intel® Hyper-Threading Technology (Intel® HT Technology) •22 MB L2 Cache; 30.25 MB L3 Cache	•2x Intel® Xeon® Processor E5-2695 v4, 2.10 GHz •18 Cores per Socket, Total 36 Physical Cores, 72 Logical Cores with Intel HT Technology •4.6 MB L2 cache; 45 MB L3 cache
Memory	•24 x 16 GB HPE SmartMemory* 1Rx4 DDR4-2666 •2 DIMMs per Channel, 6 Channels per Socket, 384 GB total DRAM	•8 x 32GB HPE SmartMemory 2Rx4 PC4-2400T-L Kit (LRDIMM – Dual-rank), •1 DIMM per Channel, 4 Channels per Socket, 256 GB Total
Local Storage	•HPE* Smart Array P408i-a SR Gen10 with RAID1 600GB 15K HDDe	•HPE* Smart Array P440ar SR Gen10 with RAID1 600GB 15K HDD
NICs	•2 x Intel® Ethernet Network Adapter XXV710 (25 GbE, SFP28) •Total: 4 Ports; 2 Ports Are Used in Tests (One Port from Each NIC).	•2 x Intel Ethernet Network Adapter XXV710 (25 GbE, SFP28) •Total: 4 Ports; 2 Ports Are Used in Tests (One Port from Each NIC).
BIOS	•HPE* U30 v1.40 (06/15/2018)	• HPE* P89 v2.42 (04/25/2017)
Software	**HPE* ProLiant* DL380 Gen10 Server**	**HPE* ProLiant* DL380 Gen9 Server**
Host Operating System	•Red Hat* Enterprise Linux* (RHEL) Server release 7.5 (Maipo) •Kernel version: 3.10.0-862.el7.x86_6	•Red Hat Enterprise Linux (RHEL) Server release 7.5 (Maipo) •Kernel version: 3.10.0-862.el7.x86_64
VM Operating System	•Ubuntu* 16.04.4 LTS (Xenial Xerus) •Kernel version: 4.4.0-133-generic x86_64	•Ubuntu* 16.04.4 LTS (Xenial Xerus) •Kernel version: 4.4.0-133-generic x86_64
KVM	•QEMU emulator version 2.11.2	•QEMU emulator version 2.11.2
Open vSwitch	•Open vSwitch* 2.9.1 release	•Open vSwitch* 2.9.1 release
DPDK	•DPDK version: 17.11.1	•DPDK version: 17.11.1
NSB	•Commit Id: 7e4b0b87	•Commit Id: 7e4b0b87
PROX	•PROX version 0.41 •Commit Id: fc6a8370	•PROX version 0.41 •Commit Id: fc6a8370
DPDK (for PROX)	•DPDK version 18.02	•DPDK version 18.02
NIC Driver	•I40e version 2.1.14-k	•I40e version 2.1.14-k
NIC Firmware	•6.02 0x80003620 1.1747.0	•6.02 0x80003620 1.1747.0

Table 5. Server Configurations

4.0 Performance Optimizations

This section describes the optimizations and tuning options used to maximize the L3 forwarding throughput of the two generation of HPE servers.

4.1 Optimize the Host

4.1.1 Isolate CPU Cores

Some of the CPU cores were isolated from the Linux scheduler to prevent the operating system (OS) from using them for housekeeping or other OS-related tasks. These isolated cores were dedicated to the Open vSwitch, DPDK PMD threads, VMs, memory banks, and the network interface card (NIC). The isolated cores were also connected to the same non-uniform memory access (NUMA) node, which helps prevent the server from using costly, cross-NUMA node links; and therefore, boosts the performance. The following commands can be executed to check the CPU configuration and the NUMA node to which a NIC belongs to.

```
# lscpu
Architecture:     x86_64
CPU op-mode(s):    32-bit, 64-bit
Byte Order:      Little Endian
CPU(s):        44
On-line CPU(s) list:  0-43
Thread(s) per core:  1
Core(s) per socket:  22
Socket(s):       2
NUMA node(s):     2
Vendor ID:       GenuineIntel
CPU family:      6
Model:         85
Model name:      Intel(R) Xeon(R) Gold 6152 CPU @ 2.10GHz
Stepping:       4
CPU MHz:        2100.000
BogoMIPS:       4200.00
Virtualization:    VT-x
L1d cache:       32K
L1i cache:       32K
L2 cache:       1024K
L3 cache:       30976K
NUMA node0 CPU(s):   0-21
NUMA node1 CPU(s):   22-43

Flags:         fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx
fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc art arch_perfmon pebs bts rep_good nopl
xtopology nonstop_tsc aperfmperf eagerfpu pni pclmulqdq dtes64 ds_cpl vmx smx est tm2 ssse3 sdbg fma cx16
xtpr pdcm pcid dca sse4_1 sse4_2 x2apic movbe popcnt tsc_deadline_timer aes xsave avx f16c rdrand lahf_lm abm
3dnowprefetch epb cat_l3 cdp_l3 intel_ppin intel_pt mba spec_ctrl ibpb_support tpr_shadow vnmi flexpriority ept
vpid fsgsbase tsc_adjust bmi1 hle avx2 smep bmi2 erms invpcid rtm cqm mpx rdt_a rdseed adx smap clflushopt clwb
xsaveopt xsavec xgetbv1 cqm_llc cqm_occup_llc cqm_mbm_total cqm_mbm_local dtherm ida arat pln pts pku ospke


# cat /sys/class/net/ens1f0/device/numa_node
0
# cat /sys/class/net/ens2f0/device/numa_node
0
# lspci | grep -i eth
12:00.0 Ethernet controller: Intel Corporation Ethernet Controller XXV710 for 25GbE SFP28 (rev 02)
12:00.1 Ethernet controller: Intel Corporation Ethernet Controller XXV710 for 25GbE SFP28 (rev 02)
37:00.0 Ethernet controller: Intel Corporation Ethernet Controller XXV710 for 25GbE SFP28 (rev 02)
37:00.1 Ethernet controller: Intel Corporation Ethernet Controller XXV710 for 25GbE SFP28 (rev 02)
```

```
# cat /sys/bus/pci/devices/0000\:12\:00.0/numa_node
0
# cat /sys/bus/pci/devices/0000\:37\:00.0/numa_node
0
```

The output of this command indicates the NUMA node number, 0 or 1, in the case of a two-socket (i.e., dual processor) system. The following commands list the associations between the CPU cores and NUMA nodes.

```
# yum install numactl

# numactl --hardware
available: 2 nodes (0-1)
node 0 cpus: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
node 0 size: 196267 MB
node 0 free: 137388 MB
node 1 cpus: 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43
node 1 size: 196607 MB
node 1 free: 133964 MB
node distances:
node  0  1
 0: 10 21
 1: 21 10
```

For the benchmark testing, all of the NICs were connected to the NUMA node 0. Hence, the CPU cores belonging to the NUMA node 0 were assigned to the Open vSwitch, DPDK PMD threads, and VMs. Table 6 shows the assignment of the CPU cores from NUMA node 0.

CPU Cores	Assigned To	Configuration Settings
0,22	OS	Set the parameters below in the /etc/default/grub file on the Linux RHEL 7.5 server to isolate cores from the kernel scheduler, and hence dedicate them to OVS-DPDK PMD threads and VMs. Cores 0 and 22 are used by the kernel, hypervisor and other host processes. `GRUB_CMDLINE_LINUX=GRUB_CMDLINE_LINUX="clearcpuid=304 default_ hugepagesz=1G hugepagesz=1G hugepages=64 hugepagesz=2M hugepages=2048 processor.max_cstate=0 intel_pstate=disable hpet=disable nosoftlockup intel_idle.max_cstate=0 mce=ignore_ce idle=poll audit=0 isolcpus=1-21,23-43 rcu_nocbs=1-21,23-43 nohz_ full=1-21,23-43 selinux=0 enforcing=0 intel_iommu=on iommu=pt crashkernel=auto rd.lvm.lv=rhel_en102/root rd.lvm.lv=rhel_en102/swap rhgb quiet"`  `Note: clearcpuid=304 is used to disable the CPU Intel® Advanced Vector Extensions 512 (Intel® AVX-512feature, resulting in higher performance. This is only applied to RHEL 7.5. Please refer to RHEL kernel-parameters.txt documentation for more details.`

2-3 (2 PMDs) or 2-5 (4 PMDs)	OVS-DPDK PMD threads	Execute the following command (mask and cores depends on scenario).
		
		`# pin PMD to cpu 2-3`
		`# ovs-vsctl set Open_vSwitch . other_config:pmd-cpu-mask=000c`
		`# pin PMD to cpu 2-5`
		`# ovs-vsctl set Open_vSwitch . other_config:pmd-cpu-mask=0003c`

CPU Cores	Assigned To	Configuration Settings

19-21 (VM1)
16-18 (VM2)
	VMs	Set the CPU core numbers for VMs in the .xml file.
		`# VM1`
		`<vcpu placement='static' cpuset='19-21'>3</vcpu>`
		`<cputune>`
		`<vcpupin vcpu='0' cpuset='19'/>`
		`<vcpupin vcpu='1' cpuset='20'/>`
		`<vcpupin vcpu='2' cpuset='21'/>`
		`</cputune>`
		`# VM2`
		`<vcpu placement='static' cpuset='16-18'>3</vcpu>`
		`<cputune>`
		`<vcpupin vcpu='0' cpuset='16'/>`
		`<vcpupin vcpu='1' cpuset='17'/>`
		`<vcpupin vcpu='2' cpuset='18'/>`
		`</cputune>`

Table 6. Sample Usage of CPU Cores in Gen10

4.1.2 Enable 1 GB Huge Pages

VMs were assigned huge pages (1 GB) to reduce translation lookaside buffer (TLB) misses by the memory management hardware and the CPU on x86_64 architecture. The following statements in the VM XML files enable 1 GB huge pages:

1. Add the following lines to the VM XML file

```
# virsh edit <vm_name>
<domain>
…
 <memoryBacking>
  <hugepages>
   <page size='1' unit='GiB' nodeset='0'/>
  </hugepages>
  <locked/>
  <nosharepages/>
 </memoryBacking>
…
```

2. Additional changes and commands (grub file, mount directory etc.).

```
# mount the hugepages
mkdir -p /dev/huge1G
mount -t hugetlbfs nodev /dev/huge1G -o pagesize=1G
mkdir -p /dev/huge2M
mount -t hugetlbfs nodev /dev/huge2M -o pagesize=2M
# add kernel boot command line parameters for hugepages
GRUB_CMDLINE_LINUX="default_hugepagesz=1G hugepagesz=1G hugepages=64 hugepagesz=2M hugepages=2048 …"
```

3. Verify the settings.

```
# cat /sys/devices/system/node/node*/meminfo | grep -i huge
Node 0 AnonHugePages:   4096 kB
Node 0 HugePages_Total:  32
Node 0 HugePages_Free:   32
Node 0 HugePages_Surp:   0
Node 1 AnonHugePages:     0 kB
```

```
Node 1 HugePages_Total:  32
Node 1 HugePages_Free:   32
Node 1 HugePages_Surp:   0

# dmesg | grep -o "default_.*"
default_hugepagesz=1G hugepagesz=1G hugepages=64 hugepagesz=2M hugepages=2048 processor.max_cstate=0
intel_pstate=disable hpet=disable nosoftlockup intel_idle.max_cstate=0 mce=ignore_ce idle=poll audit=0
isolcpus=1-21,23-43 rcu_nocbs=1-21,23-43 nohz_full=1-21,23-43 selinux=0 enforcing=0 intel_iommu=on iommu=pt
crashkernel=auto rd.lvm.lv=rhel_en102/root rd.lvm.lv=rhel_en102/swap rhgb quiet skew_tick=1

default_hugepagesz=1G hugepagesz=1G hugepages=64 hugepagesz=2M hugepages=2048 processor.max_cstate=0
intel_pstate=disable hpet=disable nosoftlockup intel_idle.max_cstate=0 mce=ignore_ce idle=poll audit=0
isolcpus=1-21,23-43 rcu_nocbs=1-21,23-43 nohz_full=1-21,23-43 selinux=0 enforcing=0 intel_iommu=on iommu=pt
crashkernel=auto rd.lvm.lv=rhel_en102/root rd.lvm.lv=rhel_en102/swap rhgb quiet skew_tick=1
```

4.1.3 Enable the Multi-Queue Feature for vhost-user and Physical DPDK Interfaces

1. For the test case with four PMDs and two queues, multiple queues were enabled in the VM XML file, and multi-queue settings were added in the following lines to change the number of queues. For information about vhost users and multi-queue, please refer to Open vSwitch documentation found at this link.

```
…
  <interface type='vhostuser'>
   <source type='unix' path='/usr/local/var/run/openvswitch/vhostuser0' mode='client'/>
   <mac address='00:04:00:00:00:01'/>
   <model type='virtio'/>
   <address type='pci' domain='0x0000' bus='0x00' slot='0x08' function='0x0'/>
   <driver queues='2'>
   <host csum='off' gso='off' tso4='off' tso6='off' ecn='off' ufo='off' mrg_rxbuf='off'/>
   <guest csum='off' tso4='off' tso6='off' ecn='off' ufo='off'/>
   </driver>
  </interface>
  <interface type='vhostuser'>
   <source type='unix' path='/usr/local/var/run/openvswitch/vhostuser1' mode='client'/>
   <mac address='00:04:00:00:00:02'/>
   <model type='virtio'/>
   <address type='pci' domain='0x0000' bus='0x00' slot='0x09' function='0x0'/>
   <driver queues='2'>
   <host csum='off' gso='off' tso4='off' tso6='off' ecn='off' ufo='off' mrg_rxbuf='off'/>
   <guest csum='off' tso4='off' tso6='off' ecn='off' ufo='off'/>
   </driver>
  </interface>
…
```

4.1.4 OVS DPDK Port/Rx Queue Assignment to PMD Threads

4.1.4.1. Two PMD Threads Test Case

1. The following commands assign Rx queues to PMD threads. The servers under test were configured for one Rx queue per port.

```
# ovs-vsctl set Interface dpdk0 options:n_rxq=1 other_config:pmd-rxq-affinity="0:2"
# ovs-vsctl set Interface dpdk1 options:n_rxq=1 other_config:pmd-rxq-affinity="0:3"
# ovs-vsctl set Interface vhostuser0 other_config:pmd-rxq-affinity="0:2"
# ovs-vsctl set Interface vhostuser1 other_config:pmd-rxq-affinity="0:3"
# ovs-vsctl set Interface vhostuser2 other_config:pmd-rxq-affinity="0:2"
# ovs-vsctl set Interface vhostuser3 other_config:pmd-rxq-affinity="0:3"
```

2. After the VM is brought up, it is possible to verify the Rx queue is pinned to the specified PMD, and the variable “isolated :” is set to ‘true’.

```
# ovs-appctl dpif-netdev/pmd-rxq-show
pmd thread numa_id 0 core_id 2:
    isolated : true
    port: dpdk0       queue-id: 0  pmd usage: 0 %
    port: vhostuser0    queue-id: 0  pmd usage: 0 %
    port: vhostuser2    queue-id: 0  pmd usage: 0 %
pmd thread numa_id 0 core_id 3:
    isolated : true
    port: dpdk1       queue-id: 0  pmd usage: 0 %
    port: vhostuser1    queue-id: 0  pmd usage: 0 %
    port: vhostuser3    queue-id: 0  pmd usage: 0 %
```

4.1.4.2. Four PMD Threads Test Case

1. The following commands assign Rx queues to PMD threads. The servers under test were configured for two Rx queues per port.

```
# ovs-vsctl set Interface dpdk0 options:n_rxq=2 other_config:pmd-rxq-affinity="0:2,1:4"
# ovs-vsctl set Interface dpdk1 options:n_rxq=2 other_config:pmd-rxq-affinity="0:3,1:5"
# ovs-vsctl set Interface vhostuser0 other_config:pmd-rxq-affinity="0:2,1:4"
# ovs-vsctl set Interface vhostuser1 other_config:pmd-rxq-affinity="0:3,1:5"
# ovs-vsctl set Interface vhostuser2 other_config:pmd-rxq-affinity="0:2,1:4"
# ovs-vsctl set Interface vhostuser3 other_config:pmd-rxq-affinity="0:3,1:5"
```

2. After the VM is brought up, it is possible to verify the Rx queue is pinned to the specified PMD and the variable “isolated :” is set to ‘true’.

```
# ovs-appctl dpif-netdev/pmd-rxq-show
pmd thread numa_id 0 core_id 2:
    isolated : true
    port: dpdk0       queue-id: 0  pmd usage: 0 %
    port: vhostuser0    queue-id: 0  pmd usage: 0 %
    port: vhostuser2    queue-id: 0  pmd usage: 0 %
pmd thread numa_id 0 core_id 3:
    isolated : true
    port: dpdk1       queue-id: 0  pmd usage: 0 %
    port: vhostuser1    queue-id: 0  pmd usage: 0 %
    port: vhostuser3    queue-id: 0  pmd usage: 0 %
pmd thread numa_id 0 core_id 4:
    isolated : true
    port: dpdk0       queue-id: 1  pmd usage: 0 %
    port: vhostuser0    queue-id: 1  pmd usage: 0 %
    port: vhostuser2    queue-id: 1  pmd usage: 0 %
pmd thread numa_id 0 core_id 5:
    isolated : true
    port: dpdk1       queue-id: 1  pmd usage: 0 %
    port: vhostuser1    queue-id: 1  pmd usage: 0 %
    port: vhostuser3    queue-id: 1  pmd usage: 0 %
```

4.1.5 Enable Core Pinning and NUMA Awareness

The following specifies the vCPU core pinning and the NUMA node tuning in VM XML file.

```
# VM1
 <vcpu placement='static' cpuset='19-21'>3</vcpu>
 <cputune>
  <vcpupin vcpu='0' cpuset='19'/>
  <vcpupin vcpu='1' cpuset='20'/>
  <vcpupin vcpu='2' cpuset='21'/>
 </cputune>
# VM2
 <vcpu placement='static' cpuset='16-18'>3</vcpu>
 <cputune>
  <vcpupin vcpu='0' cpuset='16'/>
  <vcpupin vcpu='1' cpuset='17'/>
  <vcpupin vcpu='2' cpuset='18'/>
 </cputune>
…
 <cpu mode='host-passthrough'>
  <topology sockets='1' cores='3' threads='1'/>
  <feature policy='require' name='pdpe1gb'/>
  <numa>
   <cell id='0' cpus='0-2' memory='8' unit='GiB' memAccess='shared'/>
  </numa>
 </cpu>

…
 <numatune>
  <memory mode="strict" nodeset='0'/>
 </numatune>
```

4.1.6 Performance Tuning with RHEL Tuned Profiles

RHEL provides three pre-defined, tuned profiles, called latency-performance, network-latency, and network-throughput, that can be used to improve network performance. The default is set to throughput-performance.

For all test cases, the network-latency profile was selected using the following commands to set the network latency profile. The profile is optimized for deterministic, low-latency network performance at the cost of increased power consumption.

```
# Check current tuned profile setting.
# tuned-adm active

# Set to network-latency profile.
# tuned-admin profile network-latency
```

All the system parameters used for the test cases are defined in the */usr/lib/tuned/network-latency/tuned.conf and /usr/lib/tuned/latency-performance/tuned.conf* files.

4.1.7 Linux* SMP IRQ Tuning

Each interrupt request (IRQ) has an associated affinity property, called smp_affinity, which defines the cores that are allowed to execute its interrupt service routine (ISR). Network performance can be improved by assigning non-critical IRQs to cores that are executing non-time-critical tasks, like housekeeping. The /proc/interrupts file lists IRQ numbers with the total interrupts count per core per I/O device. The /proc/irq/<IRQ_NUMBER>/smp_affinity file stores the interrupt affinity for a particular IRQ number in bit-mask format. A root user can modify this file to change an IRQ's smp_affinity.

For this test configuration, the interrupt affinity of the management NIC was initially the same as the other NICs, OVS-DPDK PMDs, and VMs cores. Changing the value of the management NIC's IRQs, using smp_affinity, resulted in higher performance.

The following example shows how to view and set the smp_affinity of a NIC interface named eno1.

```
# cat /proc/interrupts | grep eno1
581: 0 0 0 0 0 281942  0 0 0 0 … IR-PCI-MSI-edge   eno1-tx-0
582: 0 0 0 0 0 1362421  0 0 0 0 … IR-PCI-MSI-edge   eno1-rx-1
583: 0 0 0 0 0 776696  0 0 0 0 … IR-PCI-MSI-edge   eno1-rx-2
584: 0 0 0 0 0 870024  0 0 0 0 … IR-PCI-MSI-edge   eno1-rx-3
585: 0 0 0 0 0 453003  0 0 0 0 … IR-PCI-MSI-edge   eno1-rs-4

# IRQ 581 has an affinity of core 0-19
# cat /proc/irq/581/smp_affinity
000,000fffff
# set affinity to core 0 used by kernel.
# echo 1 > /proc/irq/581/smp_affinity
# cat /proc/irq/581/smp_affinity
000,00000001
```

Repeat for IRQ 582, 583, 584, and 585.

4.2 Optimize the Guest

4.2.1 Isolate CPU Cores

The following command isolates the CPU cores that will be used for DPDK PMD threads.

```
 # cat /proc/cmdline
BOOT_IMAGE=/vmlinuz-4.4.0-133-generic root=/dev/mapper/en102--ubuntu--vg-root ro default_hugepagesz=1G
hugepagesz=1G hugepages=5 isolcpus=1-2 console=tty0 console=ttyS0
```

This can be done by modifying /etc/default/grub file and then issuing the command "grub-mkconfig –o /boot/grub/grub.cfg". Reboot the VM to take effect.

4.2.2 Multiple Rx/Tx Queues Used by DPDK Application

NSB PROX Routing mode is used to simulate a VNF router in the VM. To configure it with two Rx and Tx queues per core per port, the following handle_l3fwd-2.cfg file was used to start NSB, as shown in section 5.4.

```
[eal options]
-n=6
no-output=no
[port 0]
name=if0
mac=hardware
rx desc=2048
tx desc=2048
[port 1]
name=if1
mac=hardware
```

```
rx desc=2048
tx desc=2048
[defaults]
mempool size=8K
[lua]
lpm4=dofile("ipv4-2port.lua")
[global]
start time=5
name=Routing (2x)
[core 0]
mode=master
[core 1]
name=Routing
task=0
mode=routing
route table=lpm4
rx port=if0,if1
tx port=if0,if1
drop=no
```

5.0 Scripts

This section contains scripts to set up the DPDK, Open vSwitch, and VMs, as well as to run the performance tests.

5.1 DPDK and Open vSwitch Setup

The following scripts were used to set up DPDK and Open vSwitch.

```
# Script for binding Ethernet port to igb_uio DPDK driver.
# Total 2 NICs and 1 port of each NIC are used on SUT.
#
RTE_SDK=/home/user/dpdk-stable-17.11.1
RTE_TARGET=x86_64-native-linuxapp-gcc
DPDK_TOOLS=$RTE_SDK/usertools/dpdk-devbind.py
EN1=ens1f0
EN2=ens2f0
EN1_PCI=0000:37:00.0
EN2_PCI=0000:12:00.0
ifconfig $EN1 up
ifconfig $EN2 up
cd $RTE_SDK
# Load kernel uio modules.
modprobe uio
cd $RTE_TARGET/kmod
insmod igb_uio.ko
$DPDK_TOOLS --status
$DPDK_TOOLS --bind=igb_uio $EN1_PCI $EN2_PCI
$DPDK_TOOLS -status
```

```
#
# Script for setting up OVS-DPDK with two VMs (1q/2PMDs) configuration.
# (Test Configuration #1)
#
#!/bin/bash
# OVS DPDK is configured with 6 ports (2 physical ports and 4 vhostuser ports). Each port is using 1 rx queues.
```

```
#
pkill -9 ovs
#!/bin/sh
export OVS_DIR=/home/user/openvswitch-2.9.0
export DB_SOCK=/usr/local/var/run/openvswitch/db.sock

# Load kernel openvswitch module.
modprobe openvswitch

rm -rf /usr/local/etc/openvswitch/
rm -rf /usr/local/var/run/openvswitch/

mkdir -p /usr/local/etc/openvswitch
mkdir -p /usr/local/var/run/openvswitch
mkdir -p /usr/local/var/log/openvswitch

cd $OVS_DIR
# DB creation.
./ovsdb/ovsdb-tool create /usr/local/etc/openvswitch/conf.db ./vswitchd/vswitch.ovsschema

# start ovsdb-server w/ no SSL support.
# pin to cpu 1
taskset 0x0002 ./ovsdb/ovsdb-server --remote=punix:$DB_SOCK --remote=db:Open_vSwitch,Open_vSwitch,manager_
options --pidfile --detach

# pin ovs-vswitchd on cpu 1
./utilities/ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-lcore-mask=00002

# set OVS to initialize with DPDK ports.
./utilities/ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-init=true

# allocate hugepage on socket 0.
./utilities/ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-socket-mem="4096,0"

# create ovs-vswitchd.pid file in /usr/local/var/run/openvswitch and run ovs-vswitchd as a background process.
./vswitchd/ovs-vswitchd unix:$DB_SOCK --pidfile -detach

# set max-idle value in Open_vSwitch TABLE other_config.
# The maximum time (in ms) that idle flows will remain cached in the datapath.
./utilities/ovs-vsctl set o . other_config:max-idle=50000

# pin multiple PMDs on different cpus to improve the throughput.
# pin PMD to cpu 2,3
./utilities/ovs-vsctl set Open_vSwitch . other_config:pmd-cpu-mask=000c

# create a bridge with datapath_type "netdev" in the configuration database.
./utilities/ovs-vsctl add-br br0 -- set bridge br0 datapath_type=netdev


# Add ports to Bridge.
./utilities/ovs-vsctl add-port br0 dpdk0 -- set Interface dpdk0 type=dpdk options:dpdk-devargs=0000:37:00.0
ofport_request=1
./utilities/ovs-vsctl add-port br0 dpdk1 -- set Interface dpdk1 type=dpdk options:dpdk-devargs=0000:12:00.0
ofport_request=2

# change default rxq assignment to pmd threads, rxqs may be manually pinned to desired cores.
./utilities/ovs-vsctl set Interface dpdk0 options:n_rxq=1 other_config:pmd-rxq-affinity="0:2"
```

```
./utilities/ovs-vsctl set Interface dpdk1 options:n_rxq=1 other_config:pmd-rxq-affinity="0:3"

./utilities/ovs-vsctl set Interface dpdk0 options:n_rxq_desc=2048
./utilities/ovs-vsctl set Interface dpdk1 options:n_rxq_desc=2048
./utilities/ovs-vsctl set Interface dpdk0 options:n_txq_desc=2048
./utilities/ovs-vsctl set Interface dpdk1 options:n_txq_desc=2048

# ovs server mode
# add vhostuser ports to bridge
./utilities/ovs-vsctl add-port br0 vhostuser0 -- set Interface vhostuser0 type=dpdkvhostuser ofport_request=3
./utilities/ovs-vsctl add-port br0 vhostuser1 -- set Interface vhostuser1 type=dpdkvhostuser ofport_request=4
./utilities/ovs-vsctl add-port br0 vhostuser2 -- set Interface vhostuser2 type=dpdkvhostuser ofport_request=5
./utilities/ovs-vsctl add-port br0 vhostuser3 -- set Interface vhostuser3 type=dpdkvhostuser ofport_request=6

# assign rxq to pmd threads.
./utilities/ovs-vsctl set Interface vhostuser0 other_config:pmd-rxq-affinity="0:2"
./utilities/ovs-vsctl set Interface vhostuser1 other_config:pmd-rxq-affinity="0:3"
./utilities/ovs-vsctl set Interface vhostuser2 other_config:pmd-rxq-affinity="0:2"
./utilities/ovs-vsctl set Interface vhostuser3 other_config:pmd-rxq-affinity="0:3"

# Prints a brief overview of the database contents.
./utilities/ovs-vsctl show

# Prints to the console on switch information of its flow tables and ports.
./utilities/ovs-ofctl show br0

#Clear current flows
./utilities/ovs-ofctl del-flows br0
#Add Flow for port 1 <=> 3, port 4 <=> 5, port 6 <=> 2
./utilities/ovs-ofctl add-flow br0 in_port=1,action=output:3
./utilities/ovs-ofctl add-flow br0 in_port=4,action=output:5
./utilities/ovs-ofctl add-flow br0 in_port=6,action=output:2
./utilities/ovs-ofctl add-flow br0 in_port=3,action=output:1
./utilities/ovs-ofctl add-flow br0 in_port=5,action=output:4
./utilities/ovs-ofctl add-flow br0 in_port=2,action=output:6

#display the flow table
./utilities/ovs-ofctl dump-flows br0
#display the port stats
./utilities/ovs-ofctl dump-ports br0
```

```
# Useful OVS commands to verify rx queue and PMD pinning.

ovs-appctl dpif-netdev/pmd-rxq-show
ovs-appctl dpif-netdev/pmd-stats-show

Output example (2 PMD/1q use case):
ovs-appctl dpif-netdev/pmd-rxq-show
pmd thread numa_id 0 core_id 2:
    isolated : true
    port: dpdk0       queue-id: 0  pmd usage: 31 %
    port: vhostuser0    queue-id: 0  pmd usage: 30 %
    port: vhostuser2    queue-id: 0  pmd usage: 22 %
pmd thread numa_id 0 core_id 3:
    isolated : true
```

```
    port: dpdk1       queue-id: 0  pmd usage: 32 %
    port: vhostuser1    queue-id: 0  pmd usage: 19 %
    port: vhostuser3    queue-id: 0  pmd usage: 32 %
# ovs-appctl dpif-netdev/pmd-stats-show
pmd thread numa_id 0 core_id 2:
    packets received: 5821580409
    packet recirculations: 0
    avg. datapath passes per packet: 1.00
    emc hits: 5821574204
    megaflow hits: 6166
    avg. subtable lookups per megaflow hit: 1.00
    miss with success upcall: 7
    miss with failed upcall: 0
    avg. packets per output batch: 25.22
    idle cycles: 1119629539010 (43.69%)
    processing cycles: 1443085525934 (56.31%)
    avg cycles per packet: 440.21 (2562715064944/5821580409)
    avg processing cycles per packet: 247.89 (1443085525934/5821580409)
pmd thread numa_id 0 core_id 3:
    packets received: 5766146563
    packet recirculations: 0
    avg. datapath passes per packet: 1.00
    emc hits: 5766141008
    megaflow hits: 5548
    avg. subtable lookups per megaflow hit: 1.00
    miss with success upcall: 7
    miss with failed upcall: 0
    avg. packets per output batch: 25.35
    idle cycles: 1134178045916 (44.18%)
    processing cycles: 1432812801680 (55.82%)
    avg cycles per packet: 445.18 (2566990847596/5766146563)
    avg processing cycles per packet: 248.49 (1432812801680/5766146563)
main thread:
    packets received: 0
    packet recirculations: 0
    avg. datapath passes per packet: 0.00
    emc hits: 0
    megaflow hits: 0
    avg. subtable lookups per megaflow hit: 0.00
    miss with success upcall: 0
    miss with failed upcall: 0
    avg. packets per output batch: 0.00
```

```
#
# Script for setting up OVS-DPDK with two VMs (2q/4PMDs) configuration.
# (Test Configuration #2)
#
#!/bin/bash
# OVS DPDK is configured with 6 ports (2 physical ports and 4 vhostuser ports). Each port is using 2 rx queues.
#
pkill -9 ovs
#!/bin/sh
export OVS_DIR=/home/user/openvswitch-2.9.0
export DB_SOCK=/usr/local/var/run/openvswitch/db.sock

# Load kernel openvswitch module.
```

```
modprobe openvswitch

rm -rf /usr/local/etc/openvswitch/
rm -rf /usr/local/var/run/openvswitch/

mkdir -p /usr/local/etc/openvswitch
mkdir -p /usr/local/var/run/openvswitch
mkdir -p /usr/local/var/log/openvswitch

cd $OVS_DIR
# DB creation.
./ovsdb/ovsdb-tool create /usr/local/etc/openvswitch/conf.db ./vswitchd/vswitch.ovsschema

# start ovsdb-server w/ no SSL support.
# pin to cpu 1
taskset 0x0002 ./ovsdb/ovsdb-server --remote=punix:$DB_SOCK --remote=db:Open_vSwitch,Open_vSwitch,manager_
options --pidfile --detach

# pin ovs-vswitchd on cpu 1
./utilities/ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-lcore-mask=00002

# set OVS to initialize with DPDK ports.
./utilities/ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-init=true

# allocate hugepage on socket 0.
./utilities/ovs-vsctl --no-wait set Open_vSwitch . other_config:dpdk-socket-mem="4096,0"

# create ovs-vswitchd.pid file in /usr/local/var/run/openvswitch and run ovs-vswitchd as a background process.
./vswitchd/ovs-vswitchd unix:$DB_SOCK --pidfile –detach

# set max-idle value in Open_vSwitch TABLE other_config.
# The maximum time (in ms) that idle flows will remain cached in the datapath.
./utilities/ovs-vsctl set o . other_config:max-idle=50000

# pin multiple PMDs on different cpus to improve the throughput.
# pin PMD to cpu 2,3,4,5
./utilities/ovs-vsctl set Open_vSwitch . other_config:pmd-cpu-mask=0003c

# create a bridge with datapath_type "netdev" in the configuration database.
./utilities/ovs-vsctl add-br br0 -- set bridge br0 datapath_type=netdev

# Add ports to Bridge.
./utilities/ovs-vsctl add-port br0 dpdk0 -- set Interface dpdk0 type=dpdk options:dpdk-devargs=0000:37:00.0
ofport_request=1
./utilities/ovs-vsctl add-port br0 dpdk1 -- set Interface dpdk1 type=dpdk options:dpdk-devargs=0000:12:00.0
ofport_request=2

# change default rxq assignment to pmd threads, rxqs may be manually pinned to desired cores.
./utilities/ovs-vsctl set Interface dpdk0 options:n_rxq=2 other_config:pmd-rxq-affinity="0:2,1:4"
./utilities/ovs-vsctl set Interface dpdk1 options:n_rxq=2 other_config:pmd-rxq-affinity="0:3,1:5"

./utilities/ovs-vsctl set Interface dpdk0 options:n_rxq_desc=2048
./utilities/ovs-vsctl set Interface dpdk1 options:n_rxq_desc=2048
./utilities/ovs-vsctl set Interface dpdk0 options:n_txq_desc=2048
./utilities/ovs-vsctl set Interface dpdk1 options:n_txq_desc=2048
```

```
# ovs server mode
# add vhostuser ports to bridge
./utilities/ovs-vsctl add-port br0 vhostuser0 -- set Interface vhostuser0 type=dpdkvhostuser ofport_request=3
./utilities/ovs-vsctl add-port br0 vhostuser1 -- set Interface vhostuser1 type=dpdkvhostuser ofport_request=4
./utilities/ovs-vsctl add-port br0 vhostuser2 -- set Interface vhostuser2 type=dpdkvhostuser ofport_request=5
./utilities/ovs-vsctl add-port br0 vhostuser3 -- set Interface vhostuser3 type=dpdkvhostuser ofport_request=6

# assign rxq to pmd threads.
./utilities/ovs-vsctl set Interface vhostuser0 other_config:pmd-rxq-affinity="0:2,1:4"
./utilities/ovs-vsctl set Interface vhostuser1 other_config:pmd-rxq-affinity="0:3,1:5"
./utilities/ovs-vsctl set Interface vhostuser2 other_config:pmd-rxq-affinity="0:2,1:4"
./utilities/ovs-vsctl set Interface vhostuser3 other_config:pmd-rxq-affinity="0:3,1:5"
# Prints a brief overview of the database contents.
./utilities/ovs-vsctl show

# Prints to the console on switch information of its flow tables and ports.
./utilities/ovs-ofctl show br0

#Clear current flows
./utilities/ovs-ofctl del-flows br0
#Add Flow for port 1 <=> 3, port 4 <=> 5, port 6 <=> 2
./utilities/ovs-ofctl add-flow br0 in_port=1,action=output:3
./utilities/ovs-ofctl add-flow br0 in_port=4,action=output:5
./utilities/ovs-ofctl add-flow br0 in_port=6,action=output:2
./utilities/ovs-ofctl add-flow br0 in_port=3,action=output:1
./utilities/ovs-ofctl add-flow br0 in_port=5,action=output:4
./utilities/ovs-ofctl add-flow br0 in_port=2,action=output:6

#display the flow table
./utilities/ovs-ofctl dump-flows br0
#display the port stats
./utilities/ovs-ofctl dump-ports br0
```

5.2 VM Creation

The following libvirt xml domain definition was used to create multiple VMs. It specifies huge pages memory backing, vCPU pinning, NUMA tuning, and enables multi-queues for interfaces.

```
<domain type='kvm'>
 <name>vm1-sut</name>
 <memory unit='KiB'>8388608</memory>
 <currentMemory unit='KiB'>8388608</currentMemory>
 <memoryBacking>
  <hugepages>
   <page size='1' unit='GiB' nodeset='0'/>
  </hugepages>
  <locked/>
  <nosharepages/>
 </memoryBacking>
 <vcpu placement='static' cpuset='19-21'>3</vcpu>
 <cputune>
  <vcpupin vcpu='0' cpuset='19'/>
  <vcpupin vcpu='1' cpuset='20'/>
```

```
 <vcpupin vcpu='2' cpuset='21'/>
</cputune>
<resource>
 <partition>/machine</partition>
</resource>
<os>
 <type arch='x86_64' machine='pc'>hvm</type>
 <boot dev='hd'/>
</os>
<features>
 <acpi/>
 <apic/>
</features>
<cpu mode='host-passthrough'>
 <topology sockets='1' cores='3' threads='1'/>
 <feature policy='require' name='pdpe1gb'/>
 <numa>
  <cell id='0' cpus='0-2' memory='8' unit='GiB' memAccess='shared'/>
 </numa>
</cpu>
<numatune>
 <memory mode="strict" nodeset='0'/>
</numatune>
<clock offset='utc'>
 <timer name='rtc' tickpolicy='catchup'/>
 <timer name='pit' tickpolicy='delay'/>
 <timer name='hpet' present='no'/>
</clock>
<on_poweroff>destroy</on_poweroff>
<on_reboot>restart</on_reboot>
<on_crash>destroy</on_crash>
<pm>
 <suspend-to-mem enabled='no'/>
 <suspend-to-disk enabled='no'/>
</pm>
<devices>
 <emulator>/home/user/qemu-2.11.2/bin/x86_64-softmmu/qemu-system-x86_64</emulator>
 <disk type='file' device='disk'>
  <driver name='qemu' type='qcow2'/>
  <source file='/var/lib/libvirt/images/en102-ubuntu16.04.qcow2'/>
  <backingStore/>
  <target dev='vda' bus='virtio'/>
  <alias name='virtio-disk0'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x07' function='0x0'/>
 </disk>
 <disk type='file' device='cdrom'>
  <target dev='hda' bus='ide'/>
  <readonly/>
  <alias name='ide0-0-0'/>
  <address type='drive' controller='0' bus='0' target='0' unit='0'/>
 </disk>
 <controller type='usb' index='0' model='ich9-ehci1'>
  <alias name='usb'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x05' function='0x7'/>
 </controller>
```

```
<controller type='usb' index='0' model='ich9-uhci1'>
 <alias name='usb'/>
 <master startport='0'/>
 <address type='pci' domain='0x0000' bus='0x00' slot='0x05' function='0x0' multifunction='on'/>
</controller>
<controller type='usb' index='0' model='ich9-uhci2'>
 <alias name='usb'/>
 <master startport='2'/>
 <address type='pci' domain='0x0000' bus='0x00' slot='0x05' function='0x1'/>
</controller>
<controller type='usb' index='0' model='ich9-uhci3'>
 <alias name='usb'/>
 <master startport='4'/>
 <address type='pci' domain='0x0000' bus='0x00' slot='0x05' function='0x2'/>
</controller>
<controller type='pci' index='0' model='pci-root'>
 <alias name='pci.0'/>
</controller>
<controller type='ide' index='0'>
 <alias name='ide'/>
 <address type='pci' domain='0x0000' bus='0x00' slot='0x01' function='0x1'/>
</controller>
<controller type='virtio-serial' index='0'>
 <alias name='virtio-serial0'/>
 <address type='pci' domain='0x0000' bus='0x00' slot='0x06' function='0x0'/>
</controller>
<interface type='direct'>
 <mac address='52:54:00:00:21:71'/>
 <source dev='eno1' mode='bridge'/>
 <target dev='macvtap0'/>
 <model type='virtio'/>
 <alias name='net0'/>
 <address type='pci' domain='0x0000' bus='0x00' slot='0x03' function='0x0'/>
</interface>
<interface type='vhostuser'>
 <source type='unix' path='/usr/local/var/run/openvswitch/vhostuser0' mode='client'/>
 <mac address='00:04:00:00:00:01'/>
 <model type='virtio'/>
 <address type='pci' domain='0x0000' bus='0x00' slot='0x08' function='0x0'/>
 <driver queues='1'>
 <host csum='off' gso='off' tso4='off' tso6='off' ecn='off' ufo='off' mrg_rxbuf='off'/>
 <guest csum='off' tso4='off' tso6='off' ecn='off' ufo='off'/>
 </driver>
</interface>
<interface type='vhostuser'>
 <source type='unix' path='/usr/local/var/run/openvswitch/vhostuser1' mode='client'/>
 <mac address='00:04:00:00:00:02'/>
 <model type='virtio'/>
 <address type='pci' domain='0x0000' bus='0x00' slot='0x09' function='0x0'/>
 <driver queues='1'>
 <host csum='off' gso='off' tso4='off' tso6='off' ecn='off' ufo='off' mrg_rxbuf='off'/>
 <guest csum='off' tso4='off' tso6='off' ecn='off' ufo='off'/>
 </driver>
</interface>
<serial type='pty'>
```

```
   <source path='/dev/pts/3'/>
   <target type='isa-serial' port='0'>
    <model name='isa-serial'/>
   </target>
   <alias name='serial0'/>
  </serial>
  <console type='pty' tty='/dev/pts/3'>
   <source path='/dev/pts/3'/>
   <target type='serial' port='0'/>
   <alias name='serial0'/>
  </console>
  <input type='tablet' bus='usb'>
   <alias name='input0'/>
   <address type='usb' bus='0' port='1'/>
  </input>
  <input type='mouse' bus='ps2'>
   <alias name='input1'/>
  </input>
  <input type='keyboard' bus='ps2'>
   <alias name='input2'/>
  </input>
  <graphics type='vnc' port='5901' autoport='yes' listen='10.205.205.102'>
   <listen type='address' address='10.205.205.102'/>
  </graphics>
  <sound model='ich6'>
   <alias name='sound0'/>
   <address type='pci' domain='0x0000' bus='0x00' slot='0x04' function='0x0'/>
  </sound>
  <memballoon model='virtio'>
   <alias name='balloon0'/>
   <address type='pci' domain='0x0000' bus='0x00' slot='0x0a' function='0x0'/>
  </memballoon>
 </devices>
 <seclabel type='dynamic' model='dac' relabel='yes'>
  <label>+0:+0</label>
  <imagelabel>+0:+0</imagelabel>
 </seclabel>
</domain>

# virsh commands to define, bring up, shutdown, undefine the VM
# virsh define <xml filename>
# virsh start vm1-sut
# virsh shutdown vm1-sut
# virsh undefine vm1-sut
```

A similar xml template can be used to create a second VM. Modifications may need to be incorporated such as the VM name, vcpu pinning, a new qcow image, unique mac addresses for the management interface and two vhostuser interfaces, driver queue set to 2 if it is for 2q/4PMDs test case, and a different vnc port for console access.

5.3 NSB Installation

The following script was used for NSB installation, instead of the installation steps in section 2.1 of the "Quick Start Guide for Running Yardstick*/NSB for NFVI Characterization document".

```
git clone https://gerrit.opnfv.org/gerrit/yardstick
cd yardstick

# Create a branch from commit 772d13
git checkout -b test_hp 772d13

# Pick one patch from upstream (or NSB will fail to install)
git cherry-pick 370b703896656f6

# Update DPDK to 18.02, yardstick image to 6.2.1 (more stable than "latest") and prox to fraser release
# In ./ansible/roles/download_dpdk/defaults/main.yml
        dpdk_version: "18.02"
# In ./ansible/nsb_setup.yml
        image: opnfv/yardstick:opnfv-6.2.1
# In ./ansible/roles/download_samplevnfs/defaults/main.yml
        Change "euphrates" to "fraser"

# Do not compile other sampleVNFs, as they seem not to support DPDK 18.02 in this version
# Comment out the following lines in ./ansible/ubuntu_server_baremetal_deploy_samplevnfs.yml
        #  - role: install_samplevnf
        #    vnf_name: UDP_Replay
        #  - role: install_samplevnf
        #    vnf_name: ACL
        #  - role: install_samplevnf
        #    vnf_name: FW
        #  - role: install_samplevnf
        #    vnf_name: CGNATP

sudo ./nsb_setup.sh

After entering Yardstick container (see section 2.3.1 of the Quick Start Guide), run the following

cd yardstick

# Fetch all commits
git config remote.origin.fetch +refs/heads/*:refs/remotes/origin/*
git fetch -depth=5000

# Get same yardstick version as before
git checkout -b test_hp 772d13

# Pick one (other) patch - for 25Gbps support
git cherry-pick 1d61af34eb7
```

5.3.1 Configuration Files

The following configuration files were used.

5.3.1.1. tc_prox_baremetal_l3fwd-2.yaml

Located in samples/vnf_samples/nsut/prox, this file was modified to configure the interface speed to 25 Gbps, to set the huge page size to 1GB, and to increase the total duration of the test to 5400 seconds.

```
---
schema: "yardstick:task:0.1"

scenarios:
-
 type: NSPerf
 traffic_profile: ../../traffic_profiles/prox_binsearch.yaml
 topology: prox-tg-topology-2.yaml

 nodes:
  tg__0: tg_0.yardstick
  vnf__0: vnf_0.yardstick

 options:
  interface_speed_gbps: 25
  hugepages_gb: 1
  vnf__0:
   prox_path: /opt/nsb_bin/prox
   prox_config: "configs/handle_l3fwd-2.cfg"
   prox_args:
    "-t": ""
   prox_files:
    "configs/ipv4-2port.lua" : ""
   prox_generate_parameter: True

  tg__0:
   prox_path: /opt/nsb_bin/prox
   prox_config: "configs/gen_l3fwd-2.cfg"
   prox_args:
    "-e": ""
    "-t": ""

 runner:
  type: Duration
  # we kill after duration, independent of test duration, so set this high
  duration: 5400

context:
 type: Node
 name: yardstick
 nfvi_type: baremetal
 file: prox-baremetal-2.yaml
```

5.3.1.2. handle_l3fwd-2.cfg (One Queue)

Located in samples/vnf_samples/nsut/prox/configs, this file was modified to increase the number of Rx and Tx descriptors, increase the mempool size, and use only one queue.

```
[eal options]
-n=6 ; force number of memory channels
no-output=no ; disable DPDK debug output
```

```
[port 0]
name=if0
rx desc=2048
tx desc=2048
mac=hardware
[port 1]
name=if1
rx desc=2048
tx desc=2048
mac=hardware

[defaults]
mempool size=8K

[lua]
lpm4 = dofile("ipv4-2port.lua")

[global]
start time=5
name=Routing (2x)

[core 0]
mode=master

[core 1]
name=Routing
task=0
mode=routing
route table=lpm4
rx port=if0,if1
tx port=if0,if1
drop=no
```

5.3.1.3. handle_l3fwd-2.cfg (Two Queues)

Located in samples/vnf_samples/nsut/prox/configs, this file was modified to increase the number of Rx and Tx descriptors, increase the mempool size, and use two queues.

```
[eal options]
-n=6 ; force number of memory channels
no-output=no ; disable DPDK debug output

[port 0]
name=if0
rx desc=2048
tx desc=2048
mac=hardware
[port 1]
name=if1
rx desc=2048
tx desc=2048
mac=hardware

[defaults]
```

```
mempool size=8K

[lua]
lpm4 = dofile("ipv4-2port.lua")

[global]
start time=5
name=Routing (2x)

[core 0]
mode=master

[core 1]
name=Routing
task=0
mode=routing
route table=lpm4
rx port=if0,if0
tx port=if1,if1
drop=no

[core 2]
name=Routing
task=0
mode=routing
route table=lpm4
rx port=if1,if1
tx port=if0,if0
drop=no
```

5.3.1.4. gen_l3fwd-2.cfg

Located in samples/vnf_samples/nsut/prox/configs, this file was modified to configure the number of flows to 16.

```
 [eal options]
-n=6 ; force number of memory channels
no-output=no ; disable DPDK debug output

[port 0]
name=p0
rx desc=2048
tx desc=2048
mac=hardware
[port 1]
name=p1
rx desc=2048
tx desc=2048
mac=hardware

[defaults]
mempool size=8K

[variables]
$sut_mac0=@@dst_mac0
$sut_mac1=@@dst_mac1
```

```
$tester_mac0=@@src_mac0
$tester_mac1=@@src_mac1

[global]
start time=5
name=Routing Gen

[core 0]
mode=master

[core 1]
name=p0
task=0
mode=gen
tx port=p0
bps=1875000000
pkt inline=${sut_mac0} ${tester_mac0} 08 00 45 c0 00 2e 00 01 00 00 10 11 f7 7d c1 55 01 03 0b 00 00 01 04 00
04 00 00 1a 55 7b 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11
random=0000101000000000XXXX000000000001
rand_offset=30
lat pos=42
signature pos=46
signature=0xcafedeca


[core 2]
name=p0
task=0
mode=gen
tx port=p0
bps=1875000000
pkt inline=${sut_mac0} ${tester_mac0} 08 00 45 c0 00 2e 00 01 00 00 10 11 f7 7d c1 55 01 03 0b 00 00 01 04 00
04 00 00 1a 55 7b 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11
random=0000101000000000XXXX000000000001
rand_offset=30
lat pos=42
signature pos=46
signature=0xcafedeca

[core 3]
name=p1
task=0
mode=gen
tx port=p1
bps=1250000000
pkt inline=${sut_mac1} ${tester_mac1} 08 00 45 c0 00 2e 00 01 00 00 10 11 f7 7d c1 55 01 03 0b 00 00 01 04 00
04 00 00 1a 55 7b 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11
random=0000101000000000XXXX000000000001
rand_offset=30
lat pos=42
signature pos=46
signature=0xcafedeca

[core 4]
name=p1
task=0
mode=gen
tx port=p1
bps=1250000000
```

```
pkt inline=${sut_mac1} ${tester_mac1} 08 00 45 c0 00 2e 00 01 00 00 10 11 f7 7d c1 55 01 03 0b 00 00 01 04 00
04 00 00 1a 55 7b 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11
random=0000101000000000XXXX000000000001
rand_offset=30
lat pos=42
signature pos=46
signature=0xcafedeca

[core 5]
name=REC_P0
task=0
mode=lat
rx port=p0
lat pos=42
signature pos=46
signature=0xcafedeca

[core 6]
name=REC_P1
task=0
mode=lat
rx port=p1
lat pos=42
signature pos=46
signature=0xcafedeca
```

5.3.1.5. prox-baremetal-2.yaml

Located in samples/vnf_samples/nsut/prox, this file was modified to configure the SUT and the test generator (IP addresses, username, password, PCI addresses, mac addresses...).

```
nodes:
-
  name: "tg_0"
  role: TrafficGen
  ip: 10.1.1.1
  user: "root"
  ssh_port: "22"
  password: "password"
#  key_filename: ""
  interfaces:
    xe0:
      vpci: "0000:00:08.0"
      local_mac: "48:df:37:3d:ab:9c"
      driver: "i40e"
      local_ip: "152.16.100.19"
      netmask: "255.255.255.0"
      dpdk_port_num: 0
    xe1:
      vpci: "0000:00:09.0"
      local_mac: "3c:fd:fe:aa:90:c8"
      driver: "i40e"
      local_ip: "152.16.40.19"
      netmask: "255.255.255.0"
      dpdk_port_num: 1
-
```

```
  name: "vnf_0"
  role: VNF
  ip: 10.2.2.2
  user: "root"
  ssh_port: "22"
  password: "password"
#  key_filename: ""
  interfaces:
    xe0:
      vpci: "0000:00:08.0"
      local_mac: "00:04:00:00:00:01"
      driver: "i40e"
      local_ip: "152.16.100.21"
      netmask: "255.255.255.0"
      dpdk_port_num: 0
    xe1:
      vpci: "0000:00:09.0"
      local_mac: "00:04:00:00:00:02"
      driver: "i40e"
      local_ip: "152.16.40.21"
      netmask: "255.255.255.0"
      dpdk_port_num: 1
  routing_table:
  - network: "152.16.100.20"
   netmask: "255.255.255.0"
   gateway: "152.16.100.20"
   if: "xe0"
  - network: "152.16.40.20"
   netmask: "255.255.255.0"
   gateway: "152.16.40.20"
   if: "xe1"
  nd_route_tbl:
  - network: "0064:ff9b:0:0:0:0:9810:6414"
   netmask: "112"
   gateway: "0064:ff9b:0:0:0:0:9810:6414"
   if: "xe0"
  - network: "0064:ff9b:0:0:0:0:9810:2814"
   netmask: "112"
   gateway: "0064:ff9b:0:0:0:0:9810:2814"
   if: "xe1"
```

5.3.1.6. prox_binsearch.yaml

Located in samples/vnf_samples/traffic_profiles, this file was modified to set the duration of each step to one minute and the tolerated loss to 0 percent.

```
schema: "nsb:traffic_profile:0.1"

name:      prox_binsearch
description:   Binary search for max no-drop throughput over given packet sizes

traffic_profile:
 traffic_type: ProxBinSearchProfile
 tolerated_loss: 0.0
```

```
test_precision: 0.1
packet_sizes: [64, 128, 256, 512, 1024, 1280, 1518]
duration: 60
lower_bound: 0.0
upper_bound: 100.0
```

5.4 NSB Execution

The following command was used for running the NSB task, as defined in section 2.4 of the "Quick Start Guide for Running Yardstick*/NSB for NFVI Characterization document". The prox_baremetal_l3fwd-2.yaml file is installed automatically during NSB PROX installation.

```
cd yardstick/samples/vnf_samples/nsut/prox
yardstick task start tc_prox_baremetal_l3fwd-2.yaml
```

6.0 Summary

The RFC2544 throughput results showed significant performance improvements on HPE ProLiant Gen10 servers with Intel Xeon Gold 6152 processors, compared to HPE ProLiant Gen9 servers with Intel Xeon E5-2695 v4 processors. For the RFC2544 zero packet loss test case with two VMs, two PMD threads, and one queue, the performance gain was up to 15 percent; and for the RFC2544 zero packet loss test case with two VMs, four PMD threads, and two queues, the performance gain was up to 28 percent. Increasing the number of VMs showed better improvements in server generation-to-generation performance.

Appendix A: BIOS Settings

Tables 7 and 8 show the BIOS settings that were used to enable optimal L3 forwarding performance.

Table 7. HPE ProLiant DL380 Gen10 Server (Intel Xeon Gold 6152 Processor)

Menu	Path to BIOS Setting	BIOS Option	Setting
Workload Profile			Custom
System Options	Boot Time Optimization	Dynamic Power Capping Functionality	Disabled
Processor Options	Intel ® Hyper-Threading Technology		Disabled
	Processor x2APIC Support		Enabled
Memory Options	Advanced Memory Protection		Advanced ECC Support
	Memory Refresh Rate		1x Refresh
	Channel Interleaving		Enabled
	Maximum Memory Bus Frequency		Auto
	Memory Patrol Scrubbing		Disabled
	Node Interleaving		Disabled
	Opportunistic Self-Refresh		Disabled
	Memory Remap		No Action
	Persistent Memory Options	Persistent Memory Backup Power Policy	Wait for Backup Power on Boot
		Persistent Memory Integrity Check	Enabled
		Persistent Memory Address Range Scrub	Enabled
Virtualization Options	Intel® Virtualization Technology (Intel® VT)		Enabled
	Intel® Virtualization Technology for Directed I/O (Intel® VT-d)		Enabled
	SR-IOV		Enabled

Menu	Path to BIOS Setting	BIOS Option	Setting
Power and Performance Options	Power Regulator		Static High Performance Mode
	Minimum Processor Idle Power Core C-State		No C-states
	Minimum Processor Idle Power Package C-State		No Package State
	Intel® Turbo Boost Technology		Enabled
	Energy/Performance Bias		Maximum Performance
	Collaborative Power Control		Disabled
	Intel® DMI Link Frequency		Auto
	NUMA Group Size Optimization		Clustered
	Intel® Performance Monitoring Support		Disabled
	Uncore Frequency Scaling		Maximum
	Sub-NUMA Clustering		Disabled
	Energy Efficient Turbo		Disabled
	Local/Remote Threshold		Auto
	LLC Dead Line Allocation		Enabled
	Stale A to S		Disabled
	Processor Prefetcher Options	HW Prefetcher	Enabled
		Adjacent Sector Prefetch	Enabled
		DCU Stream Prefetcher	Enabled
		DCU IP Prefetcher	Enabled
		LLC Prefetch	Disabled
		XPT Prefetcher	Auto
	I/O Options	ACPI SLIT	Enabled
		Intel® NIC DMA Channels (IOAT)	Enabled

Menu	Path to BIOS Setting	BIOS Option	Setting
		Memory Proximity Reporting for I/O	Enabled
	Intel UPI Options	Intel® UPI Link Enablement	Auto
		Intel® UPI Link Power Management	Disabled
		Intel® UPI Link Frequency	Auto
		UPI Prefetcher	Disabled
		Direct to UPI (D2K)	Auto
	Advanced Performance Tuning Options	Processor Jitter Control	Disabled
	Advanced Power Options	Redundant Power Supply Mode	Balanced Mode
Advanced Options	Mixed Power Supply Reporting		Enabled
	High Precision Event Timer (HPET) ACPI Support		Disabled
	Fan and Thermal Options	Thermal Configuration	Optimal Cooling
		Thermal Shutdown	Enabled
		Fan Installation Requirements	Enable Messaging
		Fan Failure Policy	Shutdown/Halt on Critical Fan Failures
		Extended Ambient Temperature Support	Disabled
Service Options	Memory Pre-Failure Notification		Disabled
	Advanced RAS Recovery Support		Enabled
	BDAT ACPI Support		Disabled
	Processor Monitor/Mwait Support		Disabled

Table 8. HPE* ProLiant* DL380 Gen9 Server (Intel® Xeon® Processor E5-2695 v4)

Menu	Path to BIOS Setting	BIOS Option	Setting
System Options	Processor Options	Intel ® Hyper-Threading Technology	Disabled
		Processor x2APIC Support	Enabled
	Virtualization Options	Intel® Virtualization Technology (Intel® VT)	Enabled
		Intel® Virtualization Technology for Directed I/O (Intel® VT-d)	Enabled
		SR-IOV	Enabled
	Memory Operations	Advanced memory Protection	Advanced ECC Support
Power Management	Power Profile		Maximum Performance
	Advanced Power Options	Intel® QPI Link Power Management	Disabled
		Intel® QPI Link Frequency	Auto
		Intel® QPI Link Enablement	Auto
		Energy/Performance Bias	Maximum Performance
		Maximum Memory Bus Frequency	Auto
		Channel Interleaving	Enabled
		Maximum PCI express Speed	Maximum Supported
		Dynamic Power Savings Mode Response	Fast
		Collaborative Power Control	Disabled
		Redundant Power Supply Mode	Balanced Mode
		Inte®l DMI Link Frequency	Auto
Performance Options	Intel® Turbo Boost Technology		Enabled
	ACPI SLIT		Enabled

Menu	Path to BIOS Setting	BIOS Option	Setting
	Advanced Performance Tuning Options	Node Interleaving	Disabled
		Intel® NIC DMA Channels (IOAT)	Enabled
		HW Prefetcher	Enabled
		Adjacent Sector Prefetch	Enabled
		DCU Stream Prefetcher	Enabled
		DCU IP Prefetcher	Enabled
		QPI Snoop Configuration	Early Snoop
		QPI Home Snoop Optimization	Directory + OSB Enabled
		QPI Bandwidth Optimization (RTID)	Balanced
		Memory Proximity Reporting for I/O	Enabled
		I/O Non-posted Prefetching	Enabled
		NUMA Group Size Optimization	Clustered
		Intel® Performance Monitoring Support	Disabled
Advanced Options	Fan and Thermal Options	Thermal Configuration	Increased Cooling
		Thermal Shutdown	Enabled
		Fan Installation Requirements	Enable Messaging
		Fan Failure Policy	Shutdown/Halt on Critical Fan Failures
		Extended Ambient Temperature Support	Disabled
Service Options	Processor Power and Utilization Monitoring		Disabled

Menu	Path to BIOS Setting	BIOS Option	Setting
	Memory Pre-Failure Notification		Disabled
	Shared Memory Communication		Enabled
	Memory Patrol Scrubbing		Disabled
	Memory Refresh Rate		1x Refresh
	Advanced RAS Recovery Support		Enabled
	High Precision Event Timer (HPET) ACPI Support		Disabled
	BDAT ACPI Support		Disabled
	Processor Monitor/Mwait Support		Enabled
	Primary Riser PCIe x16 Bifurcation		PCIe x16 Slot
	Hardware P-State Control		Disabled

Appendix B: Abbreviations

Abbreviation	Description
BIOS	Basic Input/Output System
DPDK	Data Plane Development Kit
NFVi	Network Functions Virtualization Infrastructure
NSB	Network Services Benchmarking
CoSP	Communications Service Provider
VNF	Virtual Network Function
PMD	Poll Mode Driver
NUMA	Non-Uniform Memory Access
NIC	Network Interface Card

Appendix C: References

#	Reference	Source
1	Intel® Open Network Platform Release 2.1 Performance Test Report SDN/NFV Solutions with Intel® Open Network Platform	https://download.01.org/packet-processing/ONPS2.1/Intel_ONP_Release_2.1_Performance_Test_Report_Rev1.0.pdf
2	Optimizing NFV Infrastructure for TCP Workloads with Intel® Xeon® Scalable Processors	https://builders.intel.com/docs/networkbuilders/optimizing-nfv-infrastructure-for-tcp-workloads-with-intel-xeon-scalable.pdf
3	Quick Start Guide for Running Yardstick*/NSB for NFV Characterization	https://wiki.opnfv.org/download/attachments/11698962/Yardstick%20NSB%20Quick%20Start%20Guide%20Final.pdf?version=1&modificationD

Legal Information

By using this document, in addition to any agreements you have with Intel, you accept the terms set forth below.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more information go to www.intel.com/benchmarks.

Optimization Notice: Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Performance results are based on testing as of September 12, 2018 and may not reflect all publicly available security updates. See configuration disclosure for details. No component or product can be absolutely secure.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request. Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Intel processors of the same SKU may vary in frequency or power as a result of natural variability in the production process.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at intel.com.

Intel® Turbo Boost Technology requires a PC with a processor with Intel Turbo Boost Technology capability. Intel Turbo Boost Technology performance varies depending on hardware, software and overall system configuration. Check with your PC manufacturer on whether your system delivers Intel Turbo Boost Technology. For more information, see http://www.intel.com/technology/turboboost.

All products, computer systems, dates and figures specified are preliminary based on current expectations, and are subject to change without notice. Results have been estimated or simulated using internal Intel analysis or architecture simulation or modeling, and provided to you for informational purposes. Any differences in your system hardware, software or configuration may affect your actual performance.

Intel® Hyper-Threading Technology requires a computer system with a processor supporting HT Technology and an HT Technology-enabled chipset, BIOS and operating system. Performance will vary depending on the specific hardware and software you use.

Intel does not control or audit third-party web sites referenced in this document. You should visit the referenced web site and confirm whether referenced data are accurate.

Intel Corporation may have patents or pending patent applications, trademarks, copyrights, or other intellectual property rights that relate to the presented subject matter. The furnishing of documents and other materials and information does not provide any license, express or implied, by estoppel or otherwise, to any such patents, trademarks, copyrights, or other intellectual property rights.

Intel, the Intel logo, Xeon, and others are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.